I/O analysis of climate applications

Arne Beer, MN 6489196, Frank Röder, MN 6526113

Introduction

About the paper and our goals

In this paper we analyze and present the strengths and weaknesses of different data structures required by some carefully picked climate and weather prediction models. Further we investigate the bare minimum of data required by those models. Another important aspect we will elaborate is the pre- and post-processing of data and the precise moment it occurs.

In the following sections we will elucidate some models and the process on how to set them up as well as running sample cases. At the end there will be a section about the overall life cycle of data.

Getting started

With intent of getting an overview about the richness of climate applications and their land, ice, weather and ocean modulation, we took an in-depth look at some models. The main goal was to find a model with proper documentation and a license which allowed us to use it for our research and benchmarking purposes. Most model investigated by us were too old for our objective. In many cases there were practically no documentation, let alone active support or new releases. The next obvious step was to get an good overview of up to date and easy to handle models which are still supported. For this purpose we put a spreadsheet together from which we could choose the most promising ones. We came across some good looking models, but despite a good first impression, most of them are shipped with broken scripts, bad documentation or a hidden provision, which forbids us to use them. After a long trial and error period of testing new models, we finally decided to stick to two models, which will be addressed later in this paper.

IFS - Integrated Forecasting System

About IFS

IFS has been chosen to be the first model for our research. *IFS* is a Model created and used by the European Center for Medium-range Weather Forecast (ECMWF). The purpose of this model is to create weather predictions by analyzing a huge amount of data. This data can be a variety of different physical bulks. (ECMWF 2016)

ECMWF offers a semi open-source version of their model for research institutions, which is called *OpenIFS*. The source code of this model can be obtained by requesting a license for the institution one is working for. They provide a good documentation about their model, which covers instruction for building, running simulations as well as very detailed information about the mathematics and techniques used in their model. After some research and two weeks passing by, we discovered a passage in their license, which forbids "Commercial and benchmarking use of *OpenIFS* models". As our original research goal is some kind of benchmarking we were forced to stop and switch to another model. We still recommend to use this model in a research or academic context, as there is plenty of documentation and a big user base.

Further progress

After the license incident with IFS we had to look for other open source models we could use for our research. We looked at many models in the following we will list some of the most promising:

- WRF(The Weather Research & Forcasting Model)
- CFS(Climate Forecast System)
- GDAS (Global Data Assimilation System)
- GFS (Global Forecast System)
- GEFS(Global Ensemble Forecast System)
- CM2.X (CM2 Global Coupled Climate Models)
- GISS GCM ModelE
- *CESM*(Community Earth System Model)
- MITgcm (M.I.T. General Circulation Model News and Information)
- GEOSCI (Geoscientific Model Development)
- Hector v1.0

- MAGICC/SCENGEN (Model for the Assessment of Greenhouse-gas Inducted Climate Change A Regional Climate SCENario GENerator)
- Metview
- COSMO(Consortium For Small Scale Modeling)
- SAGA (System for Automated Geoscientific Analyses)
- MPI-ESM
- ECOHAM

The lookup for new models took about two weeks. Most of the models mentioned above had serious flaws which forbid us to use them in our research. Many of them have stopped being maintained many years ago. In some cases there was no license provided with no available support for clarifying legal questions. In other cases it was only possible to obtain a license for our specific research goal by buying it or it was even completely forbidden to use it for benchmarking purposes, as with *OpenIFS*.

Eventually we decided to focus our work on *Community Earth System Model* (*CESM*) and Ecosystem model Hamburg Version 5 (ECOHAM5). Further we decided to take a look at *AWIPS2* a tool for displaying processed weather forecast with a very good server for handling data called *EDEX* .

Unidata - AWIPS2

AWIPS2 contains tools for weather forecast displaying and analysis. This open-source `Java` application consists of *EDEX* a data server and *CAVE* the client for data analysis and rendering. ("AWIPS II Infrastructure" 2016) Cite : "AWIPS II is a Java application consisting of a data display client *CAVE* which runs on Red Hat/CentOS Linux and OS X and a backend data server EDEX which only runs on Linux" (Unidata 2016)

EDEX (Environmental Data EXchange)

EDEX is the server for *AWIPS2* which is mainly used for preparing the data needed by *CAVE*. The *EDEX* server is a compound of different components: ("Awips System Architecture Documentation" 2016)

The first source for data is the Local Data Manager (*LDM*). This is a piece of software which shares data with computers in other networks. The Internet Data Distribution (*IDD*) provides the *LDM* with data from the Unidata community. The *LDM* can handle different kinds of data, for instance National Weather Service data streams, radar data, satellite images or grid data from numerical forecast models. The data can be directly obtained from the source or by a *LDM*

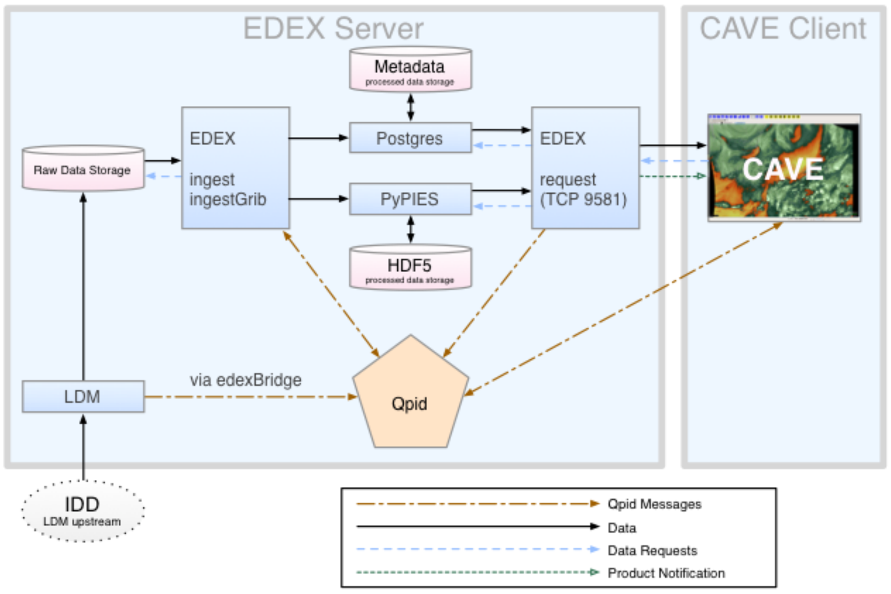


Figure 1: AWIPS2 System

that communicates with another *LDM*. When the *LDM* receives data inside the *EDEX*, a message about availability of new data is being send to the **Qipd** process, the Apache **Queue Processor Interface Daemon**, which distributes it to all other components of the *EDEX* server. The messages from **Qipd** will also contain a file header for *EDEX* to know which decoder should be used for the specific data.

After that *EDEX* can decode the data to make it ready for additional processing or signal *CAVE* that it is available for displaying. All of those messages are sent via the **edexBridge**. The default ingest server will handle all the data which are different to grib messages and is in general just responsible for the ingest of data. GRIB fully spelled **General Regularly-distributed Information in Binary form** is a data format by the ***WMO*** (World Meteorological Organization) and used for encoding results of weather models. The data is written in a binary shape into a table format. It is optimized for store and transfer data. The PostgreSQL database or **Postgres** in short is another relevant part for the storage of data in *EDEX*. It handles the metadata of the already decoded data. Postgres itself is a relational database management system which reads and store all *EDEX* metadata. The database size is not limited and can handle 32 TB of database table capacity.

HDF5 fully spelled Hierarchical Data Format (v.5) is the main format used in *AWIPS2* to store processed grids, images, etc. . Nowadays it is very similar to netCDF, which is supported by Unidata. HDF5 can handle many different types of data in a single file, for instance data of multiple radars.

The **Python Process Isolated Enhanced Storage** PyPIES has been just created for *AWIPS2* and is used for the writes and reads of data in HDF5 files. **PyPIES** is very similar in functionality compared to **Postgres**. It is a custom database abstraction layer, which processes any requests related to the HDF5 file system. The intention for this layer was to isolate the EDEX from the HDF5 processes.

CAVE (Common AWIPS Visualization Environment)

CAVE is the second part of *AWIPS2*. It is a tool for data visualization and rendering. Normally it is installed on a separated workstation apart from the other *AWIPS2* parts.
Cite: “CAVE contains of a number of different data display configurations called perspectives. Perspectives used in operational forecasting environments include D2D (Display Two-Dimensional), GFE (Graphical Forecast Editor), and NCP (National Centers Perspective). (“Awips System Architecture Documentation” 2016)

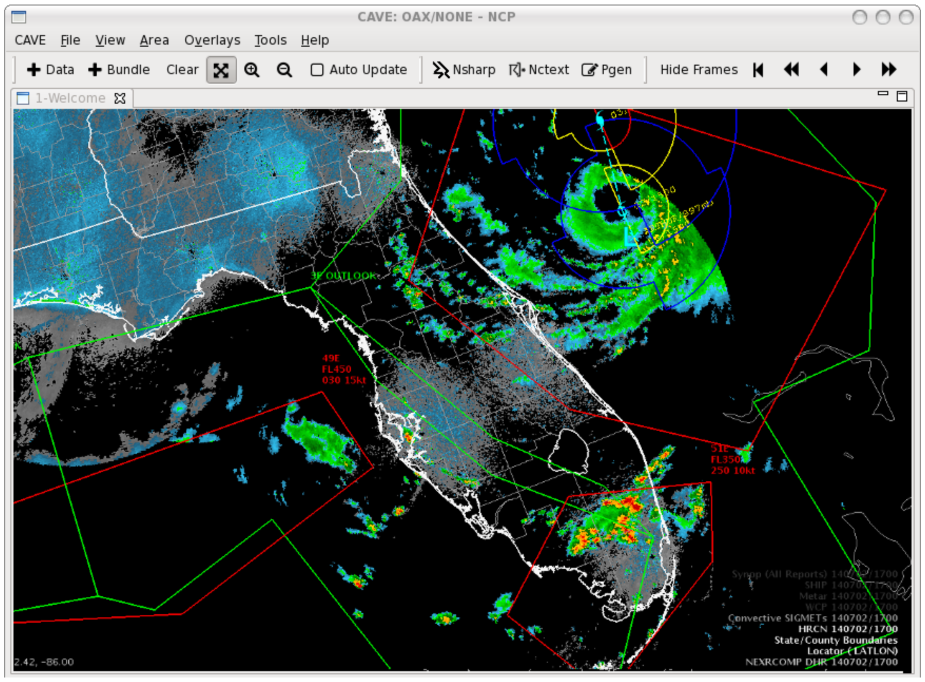


Figure 2: CAVE Example

Installation

For the installation of *AWIPS2* UniData provides a Github repository https://github.com/Unidata/awips2 and two install scripts `installCave.sh` and `installEDEX.sh`. Those install scripts use yum as a package manager and are designed for usage on CentOS, Fedora and Red Hat. To make it compatible for the *DKRZ* cluster there is more that needs to be done. As stated before *AWIPS2* is normally installed with the help of the package manager YUM and *AWIPS2* requires a directory at root location "/awips2/". There are about 2000 lines of code where "/awips2/" is hard-coded, so switching directories is not an option.

To build a version for our purpose it would be the best to have an *EDEX* on the cluster which is providing our local *CAVE* with data for visualization. Because of time constraints we were forced to move our focus away from *AWIPS2* and get back to climate models.

CESM - Community Earth System Model

About CESM

CESM itself consists of six geophysical models: ocean, land, land ice, sea ice, river and atmosphere . The *CESM* project is founded and supported by U.S. climate researchers and for the biggest part by the National Science Foundation (NSF). If there are different models in use, a so called coupler handles the time progression and overall management between the coupled models through sequences of communication. The scientific development is conducted by the *CESM* working group twice a year. For more information related to the development its recommended to visit the website **http://www.cesm.ucar.edu**. Additionally the *CESM* developers claim that it can be run out-of-the-box. "Bit-for-bit reproducibility" cannot be guaranteed, because of using different compilers and system versions.

The following *CESM* sections are referring to the userguide (Development-Team 2014).

Requirements

In the following we list some preconditions directly from the *CESM* documentation.

- UNIX style operating system such as CNL, AIX and Linux
- csh, sh, and perl scripting languages

- subversion client version 1.4.2 or greater
- Fortran (2003 recommended, 90 required) and C compilers. pgi, intel, and xlf are recommended compilers. (gfortran gcc-Version 4.8 on the cluster)
- MPI (although *CESM* does not absolutely require it for running on one processor)
- NetCDF 4.2.0 or newer. (Version 7.3 & 4.2)
- ESMF 5.2.0 or newer (optional).
- pnetcdf 1.2.0 is required and 1.3.1 is recommended (optional)
- Trilinos may be required for certain configurations X
- LAPACKm or a vendor supplied equivalent may also be required for some configurations. (Version 3.0)
- CMake 2.8.6 or newer is required for configurations that include CISM. (Version 2.8.12.2 on the cluster)

Installation

- Open source
- Download at **CCMS(Click here)**:
  - Username: guestuser
  - Password: friendly
- Version 1.2.1
- Available with svn:

  `svn co`

  https://svn-ccsm-models.cgd.ucar.edu/cesm1/release_tags/cesm1_2_1 cesm1_2_1 –username guestuser –password friendly
- We recommend to create an entry in your `~/.subversion/servers` config for later svn usage with scripts:

```
[groups]
cesm = svn-ccsm-inputdata.cgd.ucar.edu

[cesm]
username = guestuser
store-passwords = yes
```

Most parts of the *CESM* software project are open source. However three libraries are published by the Los Almos National Laboratory, who licensed their software as free to use as long as it isn't used in a commercial context. Affected libraries are POP, SCRI and CICE (***For link to license click here***).

Input Data Set

Setup

There is actually a set of input data which can be downloaded and configured for *CESM*. It can be made available through another subversion input data repository using the same user name as used in the installation above.
The dataset is around 1 TB big and should not be downloaded entirely. The download is regulated on demand, which means if *CESM* needs the particular data it will be downloaded and checked automatically by *CESM* itself. The data should be on a device with a fast connection to the actual computing device.
The data will be downloaded into the `$DIN_LOC_ROOT` folder, which has to be set in the `env_run.xml` in the "Build Setup" step later on. Multiple users can use the same `$DIN_LOC_ROOT` directory and it should thereby be configured as group writable.
The input data can be downloaded manually as well as using the `check_input_data` script. The script triggers a partial download of the svn input data repository and allows to exactly specify the data that should be downloaded. This script is called during the `setup_case` step as well. If the specified data is not found in `$DIN_LOC_ROOT` it will automatically be downloaded by the script.
If one likes to download the input manually it should be done **before** building *CESM*. In addition it is also possible to download the data via svn subcommands direct, but it is much better to use the `check_input_data` script as it secures to download only the required data.
If the machine is supported by *CESM*, there should be a preset in `ccsm_utils/Machines`. Otherwise there is the possibility to make it run on generic machines with the variable `-mach userdefined` as argument for the `./scripts/create_newcase` script and further configuration afterwards.

CESM Creating And Configure A New Case

Prerequisites

`perl-switch` and `csh` is needed for the further project setup.

Create a new case

Cases are created with a resolution and configurations fitting to the machine it should be executed at. To create a case execute `./scripts/create_newcase` with the respective parameters. This is a command for a user defined machine with the B1850CN input data set:

```
# Parameters are respectively:
# Case name and directory location
# Machine name
# Component set name
# Resolution
/create_newcase -case ./test1 \
    -mach userdefined \
    -compset B1850CN  \
    -res f45_g37
```

In the original repository many errors occur due to deprecated syntax and buggy setup code. We recommend to use our updated version of the code. The text `Successfully created the case` should appear on your screen. If problems with `create_newcase` occur, one should try one of the examples listed in the error message.

In case `create_newcase` breaks while calling one of the `mkbatch.*` scripts, you probably need to install CShell, as those scripts are written for `#!/bin/csh`.

The result of `create_newcase` is a directory `.../cesm/scripts/<YourCase>` with a bunch of directories or filenames to be explained:

```
README.case - This files will contain tracked problems
        and changes at runtime
CaseStatus  - A File containing a history of operations
         done in the actual case
BuildConf/  - The files in this directory are scripts
        for generating component
        name lists and utility libraries. They should
         never be edited.
SourceMods/ - This directory is for modified source code
LockedFiles/    - It contains copies of files that should not
         be changed, xml are locked until the clean
        operation is executed
Tools/      - A directory which contains support scripts.
         They should never be edited.
env_mach_specific   - machine-specific variables for
          building/running are set here
env_case.xml    - Case specific variables like the root
        and models are set(cannot be changed, have
        to re-run create_newcase for changes)
env_build.xml   - Contains the build settings, resolution and
         configuration options
env_mach_pers.xml   - Sets the machine processor layout
env_run.xml - Contains run-time settings
```

```
cesm_setup  - Script for set up
$CASE.$MACH.build   - Script for building components,
     executables and utility libraries
$CASE.$MACH.clean_build - Remove all object files
                and libraries
$CASE.$MACH.l_archive   - Script for long-term archiving of
    output (only if it is available on the machine)
xmlchange   - Utility to change values in other .xml files
preview_namelists   - Utility to see the component name lists
check_input_data    - Check for input datasets
check_production_test   - Creates a test of the owners case
```

Setup case

Once a case has been created by the previous command, the setup has to be completed. To achieve this, the `cesm_setup` script in the case directory needs to be executed. Settings for this specific case are specified in `env_mach_pes.xml`. The documentation states that this file should only be manipulated by using the `xmlchange` script. As we want to use our own machine, we need to create a user defined machine for this test case.

Values that need to be set:

- `MAX_TASKS_PER_NODE` in `env_mach_pes.xml`
- `OS` in `env_build.xml`
- `MPILIB` in `env_build.xml`
- `COMPILER` in `env_build.xml`
- `EXEROOT` in `env_build.xml`
- `RUNDIR` in `env_run.xml`
- `DIN_LOC_ROOT` in `env_run.xml`

There is an example configuration in `scripts/example_config`. This configuration expects a folder in root `/cesm` and `/cesm/inputdata`, but if you don't have root access at your location, those variables can be easily changed (`EXEROOT`, `RUNDIR`, `DIN_LOC_ROOT`).

Build case

To build a case, the `$CASENAME.build` script needs to be executed. In case you chose the `gnu` compiler in your settings, make sure you have `gmake` installed and create a symlink from `gmake` to `make`. If any previous builds failed, `$CASENAME.clean_build` needs to be executed.

Getting data

If you don't want to download input data manually, jump to the `Quickstart` chapter. The data download script lies directly in `cesm1_2_1/scripts/yourcase` you created one step back. To download input data to a specific data directory execute this with an adjusted path.

```
 export DIN_LOC_ROOT='/Path/to/input/data/dir'
 mkdir -p $DIN_LOC_ROOT
./check_input_data -inputdata $DIN_LOC_ROOT
-export -datalistdir $DIN_LOC_ROOT
```

Now it also should be possible to check if the required data is present with follow command:

```
check_input_data -inputdata $DIN_LOC_ROOT -check
```

To download missing data from the server use:

```
check_input_data -inputdata $DIN_LOC_ROOT -export
```

Booth commands need to be run inside the `$CASEROOT`

Build the Case

```
cd ~/cesm/EXAMPLE_CASE
./cesm_setup
./EXAMPLE_CASE.build
```

Quickstart

This QuickStart (Development-Team 2014) should give an brief overview about the work flow of *CESM* , especially if there already is a version ported to the local target machine. If that is not the case, start with the more detailed description above.

There are a couple of definitions which have to be kept in mind:

```
$COMPSET refers to the components set
$RES refers to the model resolution
$MACH refers to the target machine
$CCSMROOT refers to the _CESM_ root directory
$CASE refers to the case name
$CASEROOT refers to the full pathname of the root directory
      where the case ($CASE) will be created
$EXEROOT refers to the executable directory
      ($EXEROOT is normally __not__ the same as $CASEROOT)
$RUNDIR refers to the directory where _CESM_ actually runs.
```

```
This is normally set to $EXEROOT/run.
(changing $EXEROOT does not change $RUNDIR
as these are independent variables)
```

In the first step you need to ***download(Click here)*** *CESM* and select a machine, a component set and a resolution from the list displayed after using this commands:

```
> cd $CCSMROOT/scripts
> create_newcase -list
```

There is a list of *CESM* supported components like ***sets(Click here)***, ***resolution(Click here)*** and ***machines(Click here)***. Remember, that the `-list` will always provide a list of supported component sets for the local *CESM* version. The first letters of the `-compset` option will indicate which kind of model is used. To create a case the command `create_newcase` is used. It creates a case directory containing the scripts and XML files to set up the configurations for resolution, component set and machine requested. The `create_newcase` has some arguments as condition and some additional options for generic machines. For more information `create_newcase -h` should help. In case that a supported machine is in use `($MACH)` type the following words:

```
>create_newcase -case $CASEROOT \
        -mach $MACH \
        -compset $COMPSET \
        -res $RES
```

When using the machine setting `userdefined` it is required to edit the resulting `xml` files and fill them with the informations needed for the target machine. The `create_newcase -list` command will also show all available machines for the local version. For running a new target machine use the **section above**.

To setup the `case run script` be sure to use the `cesm_setup` command which creates a $CASEROOT$/CASE.run script with `user_nl_xxx` files, while the xxx tell us something about the case configuration. But before running `cesm_setup` there is the `env_mach_pes.xml` file in $CASEROOT to be modified for the experiment.

```
> cd $CASEROOT
```

After this the `env_mach_pes.xml` can be modified with the ***xmlchange*** command. Take a look at `xmlchange -h` for detailed information. Then the `cesm_setup` can be initiated.

```
> ./cesm_setup
```

With the optional build modifications in mind (`env_mach_pes.xml`) the build script can be startet:

```
> $CASE.build
```

To run the case and maybe setting the variable $DOUT_S in `env_mach_pes.xml` to `false` the job can be submitted to the batch queue:

```
> $CASE.submit
```

After the job finished you can review all the following directories and files like:

```
1. $RUNDIR
  * the directory set in the `env_build.xml` file
  * the location where the _CESM_ was run with log files for
every part
2. $CASEROOT/logs
  * if the run was successful the log files have been copied
into this directory
3. $CASEROOT
  * here should a standard out or error file
4. CASEROOT/CaseDocs
  * a list a case names is copied to this directory
5. CASEROOT/timing
    * here are timing files which are representing the
  performance of the model
  6. $DOUTS_S_ROOT/$CASE
    * This directory is an archive depending on the setting
  done above, while it is true there is a log and history
```

Conclusion

We managed to fix the build scripts to create and setup a case. In the step of compiling the code we encountered a few errors, but we were able to compile two models. After that another error occurred during compilation of the `pio` module. *CESM* ships with a parallel IO library, which practically is a set of interfaces for netcdf, parallel netcdf or binary IO. We chose to use pnetcdf for our build. After installing the required libraries and setting the proper paths and variables, as described in their documentation, the build still failed and required a configuration file for `pio`. There is no further information about this configuration file in their documentation. After writing their support without response, we were forced to stop using *CESM*.

After all *CESM* still looks like a promising model, but it has many flaws. Their build scripts aren't generic. Nearly 6 weeks were spent to understand the code and fix old syntax or hard-coded paths. If there were better documentation about the setup and compilation for *CESM* we could've probably used it, as we were really close to compiling it.

ECOHAM5

About

ECOHAM5 (ECOsystem Model Hamburg Version 5) is a physical-geological-biochemical model for different levels of ocean depth in which different elements react with each other. This version takes advantage of parallelism through interprocess communication. The physic behind this model is based on the hydrodynamic model called *HAMSOM*. In comparison to older *ECOHAM5* versions this one has a better generic approach to cope with different grid resolutions. The information in this section were taken from the "ECOHAM5 user guide" (Fabian Große last updated: November 27, 2015).

Source Code

Not freely available in the internet.

Compile ECOHAM5

To compile *ECOHAM5* for a testcase, there is the following script to be run in different ways:

```
./CompileJob-cluster.sh TEST 0  // for just the compiling

./CompileJob-cluster.sh TEST 1
    // for compiling and make it ready to run

./CompileJob-cluster.sh TEST 2  // compiling and run the model
```

In our case `TEST` is the data input. If an error occurs and you don't have the permission to run `sbatch` on a specified partitions, it is necessary to make modification in the `RunJob.TEST` located in the folder of the declared input. The `wrk` directory will contain most of the output generated by the case. The directory `/Input` inside the `/wrk` folder contain all the input used for the case. There are `.dat` `.header` and `.direct` files which are providing the application with all data needed. To visualize the newly created case there is a folder named `/res.TEST` with a subdirectory `/TEST.1977.00` in which you can find the netCFD data file `TEST_3D.nc`. For our purpose we used **ncview** and **ncdump** to analyze the generated output.

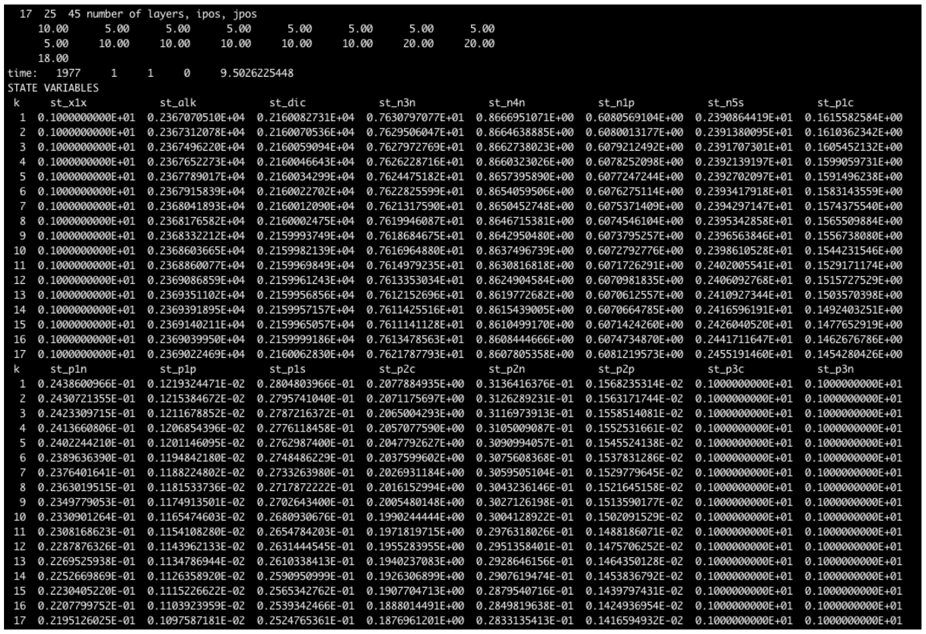
Using ncview

Ncview is very easy to start once you know where the `.nc` file of your choice is. For help beyond our purpose in this paper, we recommend to have a look at their homepage http://cirrus.ucsd.edu/....

Figure 2 shows Europa and the wind speed colored by a scale. There are a bunch of settings which can manipulate the view at the data. An example of the resulting insight can be seen in *Figure 2*.

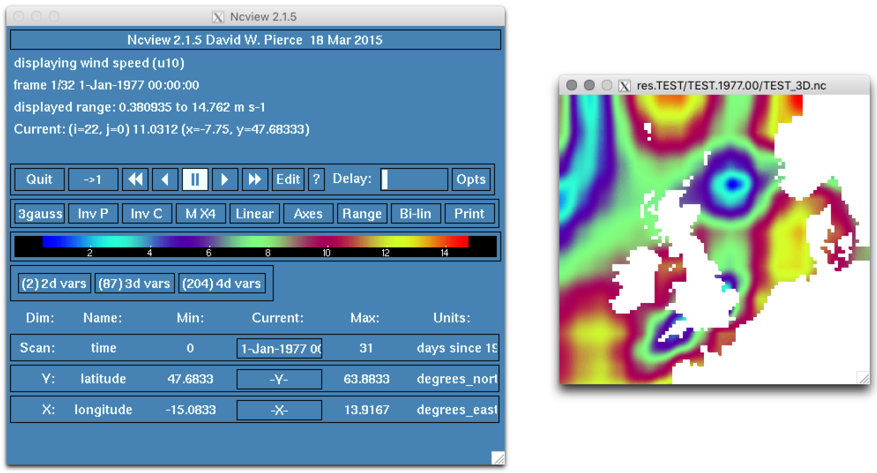
Using ncdump

Ncdump has a different approach of reviewing data, by converting a netCDF data file in to a better readable text format for humans. A result of ncdump can be seen in *Figure 3*.



```
  17  25  45 number of layers, ipos, jpos
    10.00      5.00      5.00      5.00      5.00      5.00      5.00      5.00
     5.00     10.00     10.00     10.00     10.00     10.00     20.00     20.00
    18.00
time:   1977     1     1     0     9.5026225448
STATE VARIABLES
 k     st_x1x           st_alk           st_dic           st_n3n           st_n4n           st_n1p           st_n5s           st_p1c
  1  0.1000000000E+01  0.2367070510E+04  0.2160082731E+04  0.7630797077E+01  0.8666951071E+00  0.6080569104E+00  0.2390864419E+01  0.1615582584E+00
  2  0.1000000000E+01  0.2367312078E+04  0.2160070536E+04  0.7629506047E+01  0.8664638885E+00  0.6080013177E+00  0.2391380095E+01  0.1610362342E+00
  3  0.1000000000E+01  0.2367496220E+04  0.2160059094E+04  0.7627972769E+01  0.8662738023E+00  0.6079212492E+00  0.2391707301E+01  0.1605452132E+00
  4  0.1000000000E+01  0.2367652273E+04  0.2160046643E+04  0.7626228716E+01  0.8660323026E+00  0.6078252098E+00  0.2392139197E+01  0.1599059731E+00
  5  0.1000000000E+01  0.2367789017E+04  0.2160034299E+04  0.7624475182E+01  0.8657395890E+00  0.6077247244E+00  0.2392702097E+01  0.1591496238E+00
  6  0.1000000000E+01  0.2367915839E+04  0.2160022702E+04  0.7622825599E+01  0.8654059506E+00  0.6076275114E+00  0.2393417918E+01  0.1583143559E+00
  7  0.1000000000E+01  0.2368041893E+04  0.2160012090E+04  0.7621317590E+01  0.8650452748E+00  0.6075371409E+00  0.2394297147E+01  0.1574375540E+00
  8  0.1000000000E+01  0.2368176582E+04  0.2160002475E+04  0.7619946087E+01  0.8646715381E+00  0.6074546104E+00  0.2395342858E+01  0.1565509884E+00
  9  0.1000000000E+01  0.2368332212E+04  0.2159993749E+04  0.7618684675E+01  0.8642950480E+00  0.6073795257E+00  0.2396563846E+01  0.1556738080E+00
 10  0.1000000000E+01  0.2368603665E+04  0.2159982139E+04  0.7616964880E+01  0.8637496739E+00  0.6072792776E+00  0.2398610528E+01  0.1544231546E+00
 11  0.1000000000E+01  0.2368860077E+04  0.2159969849E+04  0.7614979235E+01  0.8630816818E+00  0.6071726291E+00  0.2402005541E+01  0.1529171174E+00
 12  0.1000000000E+01  0.2369086859E+04  0.2159961243E+04  0.7613353034E+01  0.8624904584E+00  0.6070981835E+00  0.2406092768E+01  0.1515727529E+00
 13  0.1000000000E+01  0.2369351102E+04  0.2159956856E+04  0.7612152696E+01  0.8619772682E+00  0.6070612557E+00  0.2410927344E+01  0.1503570398E+00
 14  0.1000000000E+01  0.2369391895E+04  0.2159957157E+04  0.7611425516E+01  0.8615439005E+00  0.6070664785E+00  0.2416596191E+01  0.1492403251E+00
 15  0.1000000000E+01  0.2369140211E+04  0.2159965057E+04  0.7611141128E+01  0.8610499170E+00  0.6071424260E+00  0.2426040520E+01  0.1477652919E+00
 16  0.1000000000E+01  0.2369039950E+04  0.2159999186E+04  0.7613478563E+01  0.8608444666E+00  0.6074734870E+00  0.2441711647E+01  0.1462676786E+00
 17  0.1000000000E+01  0.2369022469E+04  0.2160062830E+04  0.7621787793E+01  0.8607805358E+00  0.6081219573E+00  0.2455191460E+01  0.1454280426E+00
 k     st_p1n           st_p1p           st_p1s           st_p2c           st_p2n           st_p2p           st_p3c           st_p3n
  1  0.2438600966E-01  0.1219324471E-02  0.2804803966E-01  0.2077884935E+00  0.3136416376E-01  0.1568235314E-02  0.1000000000E+01  0.1000000000E+01
  2  0.2430721355E-01  0.1215384672E-02  0.2795741040E-01  0.2071175697E+00  0.3126289231E-01  0.1563171744E-02  0.1000000000E+01  0.1000000000E+01
  3  0.2423309715E-01  0.1211678852E-02  0.2787216372E-01  0.2065004293E+00  0.3116973913E-01  0.1558514081E-02  0.1000000000E+01  0.1000000000E+01
  4  0.2413660806E-01  0.1206854396E-02  0.2776118458E-01  0.2057077590E+00  0.3105009087E-01  0.1552531661E-02  0.1000000000E+01  0.1000000000E+01
  5  0.2402244210E-01  0.1201146095E-02  0.2762987400E-01  0.2047792627E+00  0.3090994057E-01  0.1545524138E-02  0.1000000000E+01  0.1000000000E+01
  6  0.2389636390E-01  0.1194842180E-02  0.2748486229E-01  0.2037599602E+00  0.3075608368E-01  0.1537831286E-02  0.1000000000E+01  0.1000000000E+01
  7  0.2376401641E-01  0.1188224802E-02  0.2733263980E-01  0.2026931184E+00  0.3059505104E-01  0.1529779645E-02  0.1000000000E+01  0.1000000000E+01
  8  0.2363019515E-01  0.1181533736E-02  0.2717872222E-01  0.2016152994E+00  0.3043236146E-01  0.1521645158E-02  0.1000000000E+01  0.1000000000E+01
  9  0.2349779053E-01  0.1174913501E-02  0.2702643400E-01  0.2005480148E+00  0.3027126198E-01  0.1513590177E-02  0.1000000000E+01  0.1000000000E+01
 10  0.2330901264E-01  0.1165474603E-02  0.2680930676E-01  0.1990244444E+00  0.3004128922E-01  0.1502091529E-02  0.1000000000E+01  0.1000000000E+01
 11  0.2308168623E-01  0.1154108280E-02  0.2654784203E-01  0.1971819715E+00  0.2976318026E-01  0.1488186071E-02  0.1000000000E+01  0.1000000000E+01
 12  0.2287876326E-01  0.1143962133E-02  0.2631444545E-01  0.1955283955E+00  0.2951358401E-01  0.1475706252E-02  0.1000000000E+01  0.1000000000E+01
 13  0.2269525938E-01  0.1134786944E-02  0.2610338413E-01  0.1940237083E+00  0.2928646156E-01  0.1464350128E-02  0.1000000000E+01  0.1000000000E+01
 14  0.2252669869E-01  0.1126358920E-02  0.2590950999E-01  0.1926306899E+00  0.2907619474E-01  0.1453836792E-02  0.1000000000E+01  0.1000000000E+01
 15  0.2230405220E-01  0.1115226622E-02  0.2565342762E-01  0.1907704713E+00  0.2879540716E-01  0.1439797431E-02  0.1000000000E+01  0.1000000000E+01
 16  0.2207799752E-01  0.1103923959E-02  0.2539342466E-01  0.1888014491E+00  0.2849819638E-01  0.1424936954E-02  0.1000000000E+01  0.1000000000E+01
 17  0.2195126025E-01  0.1097587181E-02  0.2524765361E-01  0.1876961201E+00  0.2833135413E-01  0.1416594932E-02  0.1000000000E+01  0.1000000000E+01
```

Figure 3: ECOHAM5 with ncdump



Ncview 2.1.5 David W. Pierce 18 Mar 2015

displaying wind speed (u10)

frame 1/32 1-Jan-1977 00:00:00

displayed range: 0.380935 to 14.762 m s-1

Current: (i=22, j=0) 11.0312 (x=-7.75, y=47.68333)

Dim:	Name:	Min:	Current:	Max:	Units:
Scan:	time	0	1-Jan-1977 0(	31	days since 19
Y:	latitude	47.6833	-Y-	63.8833	degrees_nort
X:	longitude	-15.0833	-X-	13.9167	degrees_east

Figure 4: Visualization of ECOHAM5 output data in ncview visual browser

Life cycle of data

General

Through the whole process of running a simulation there are different types of data at certain points. The complexity and the information can differ. The data which is fed into the program at the beginning won't be the same which is visualized by a color on the climate overview. (University 2016)

The life cycle could be divided into the parts shown in *Figure 4* ("The Data Lifecycle" 2016) and can differ from institution to institution:

1. creating data
2. processing data (pre- and post-processing included)
3. analyzing data
4. preserving data
5. giving access to data
6. re-using data

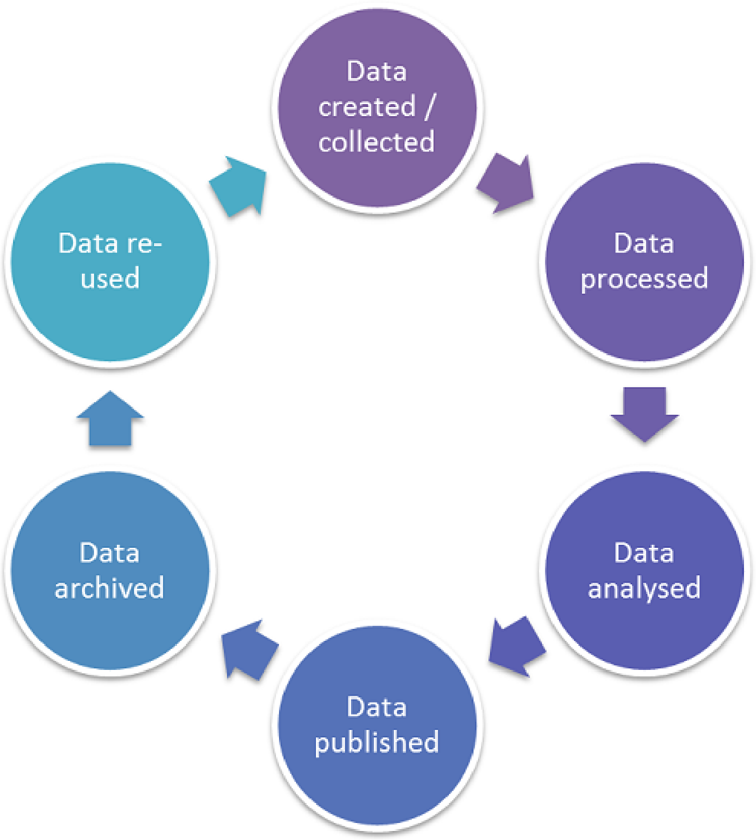


Figure 5: Data life cycle

Creating the data

Creating the data could also be referred to as the design of the research. The choices made during this step will be crucial, because it will have an huge impact on the processing as well as at the overall structure of the application. Which kind of data management, formats or storage should be used are just some examples of questions which needs to be answered for a proper design. If there are already similar existing simulations, their data could be re-used for the current step. In case there is no similar simulation, new data needs to be collected by research and experiments. Running simulations and capturing metadata are important parts of this process.

Processing data

This step will contain the digitization, transcribing and translation of data into a useful figure. It also is about the vetting of validate and clean data. Anonymizing data could also be part of processing. Sometimes there is pre- and post-processing besides the general processing. Pre-processing will prepare the particular data before a certain step. In this step all unneeded information will be dismissed to save time and storage capacity. After this there might be the post-processing. This process includes the treatment of data to make it usable for following steps. For example the data should be described and explained to make it easy to read for other people. As a last step it is necessary to take a look at the storage of the data.

Analyzing of data

It is hard to get any valuable information out of pure data, especially if there are thousands of values and rows. Therefore we need methods to interpret them. Derivation of data during calculation of the output of the research and making publications is another part. Now main calculations of the data are done, the data have to be prepared for preservation.

Preserving of data

Preservation of data is about looking for the best formats and best media on which it should be backed up and stored. Also creation of meta data and documentation is important for the final archiving.

Giving access to data

Because research is done mostly by public institutes there is the demand of sharing and distributing the data and knowledge. Considering a mechanism for access control and determine copyrights might also be very useful.

Re-using data

Research in the future could also be based on the work which was done in the past. Teaching and publications may be good examples for re-using.

Summary

With the growth of data in simulations and the constant rapid improvement of processing power, it is very essential for research establishments to maintain a proper storage technology. The data and knowledge are the key and the only thing those institutes are working for. Therefore data and knowledge needs the right treatment.

Conclusion

This project was about the analysis of input and output of climate applications. We made our way through running models provided by our supervisors and some we found in the internet which seemed suitable for our purpose. We faced bad documentations and insufficient scripts for making such a model run. At the end we didn't manage to run most of them at different levels of progress. *ECOHAM5* was the only one which worked on the fly Apart of that it didn't kept us away from documenting our respective progress on the models as well as analyzing the life cycle of data in general.

References

"AWIPS II Infrastructure." 2016. http://www.unidata.ucar.edu/software/awips2/images/awips2_coms.png.

"Awips System Architecture Documentation." 2016. https://docs.google.com/document/d/1ggqsioNSwVC6XzJL7oJs1vPkUPzXeaCwz25tMtqikzE/edit.

Development-Team, CESM1.2. 2014. "Cesm Documentation." http://www.cesm.ucar.edu/models/cesm1.2/cesm/doc/usersguide/book1.html.

ECMWF. 2016. "OpenIFS documentation." https://software.ecmwf.int/wiki/display/OIFS/About+OpenIFS.

Fabian Große, Hermann Lenhart, Markus Kreus. last updated: November 27,

2015. “ECOHAM5 user guide.” handed via mail.

“The Data Lifecycle.” 2016. http://www.lancaster.ac.uk/media/lancaster-university/content-assets/images/library/other/DataLifecycle.png.

Unidata. 2016. “Unidata AWIPS II, Softwarepage.” https://www.unidata.ucar.edu/software/awips2/.

University, Lancaster. 2016. “The Data Lifecycle.” http://www.lancaster.ac.uk/library/rdm/what-is-rdm/plan/data-lifecycle/.